Undirected Connectivity is in **L**
an overview of the proof

A CS254 final project by [redacted] and [redacted]

March 15, 2019

1 Introduction

Suppose you want to see if you can get from Point s to Point t in a maze (i.e. an undirected graph) — but you have terrible memory. What are your options? One idea is to walk around randomly from s until you find t or run out of patience and declare that t is 'unreachable.' Then you only need to remember where you currently are. This is a small amount of memory compared to an algorithm like DFS, where you also need to remember where you came from.

On what graphs does this algorithm work well? It works well on graphs that are 'well-connected,' so that for any s and t a random walk from s finds t with very high probability. Similarly you want your graph to be 'sparse' so that you don't have too many choices at each step. Graphs that are both 'sparse' and 'well-connected' are called 'expander graphs.' A well-designed train system should be an expander graph, in the sense that we don't spend too many resources building train lines from every city to every other city, but you can nonetheless get between cities without too much trouble. Scientists theorize that the neurons in your brain also form expander graphs, because even though each neuron has just a few neighbors, signals can spread quickly across your brain. We give a more formal definition of an expander graph in Section 2, along with two more equivalent definitions that will be useful for the rest of the paper.

It turns out that a randomly-generated graph is *likely* to be an expander graph... but of course not *all* graphs are expanders. Okay, but perhaps we can somehow convert an existing graph into an expander graph? As long as we retain s-t-connectivity, this should only help our chances of finding t. In Section 3, we will explore an extremely clever construction that does exactly this: it improves the expansion of a graph while preserving s-t-connectivity. This construction repeats two steps over and over again: the first step, *squaring*, improves the 'well-connected-ness' property at the cost of the 'sparse-ness' property; the second step, the *zig-zag product*, then recovers the 'sparse-ness' without sacrificing the gains on 'well-connected-ness.'

To summarize, the square-then-zig-zag construction preserves s-t-connectivity in our 'seed' graph, but also makes it more 'well-connected.' Thus, a short (indeed, logarithmic-length) random walk on this new graph suffices to detect if you can reach s from t. Ah, but we can simply brute-force through all paths of logarithmic length using logarithmic space! Section 4 fills in the technical details for this proof. This recent result, due to Omer Reingold (2008), places USTCON in the complexity class **L** and is an important theoretical achievement.

2 Some Definitions of Expander Graphs

Our goal is to measure 'well-connectedness' in a useful way. In this section, we will start with a 'combinatorial' definition of well-connectedness. Then, we will develop a completely different, 'linear-algebraic' definition of well-connectedness, and assert that they are indeed equivalent. While the former measure is more natural, the latter is easier to compute.

2.1 A natural measure of well-connectedness

Suppose $G = \langle V, E \rangle$ is an undirected multi-graph. In particular, there can be more than one edge between any pair of vertices — including a vertex and *itself*. Hence, we can think of E as a function $E : (V \times V) \to \mathbb{N}$.

Definition 2.1. *G is a (K, A)* expander graph *if*

1. *G is* sparse. *For some integer D, the degree of each vertex is exactly D. We call such graphs D-regular.*

2. *G is* well-connected. *For any subset $S \subseteq V$ of size at most K, let $S' = V \setminus S$. Then, the number of edges $e = (s, s')$ such that $s \in S$ and $s' \in S'$ is at least $A \cdot |S|$.*

[Typically $K = \Omega(|V|), A = \Theta(D)$.]

Intuitively, we can think of this as saying that we can 'escape' any small (size-K) subgraph of the graph easily (with high probability) by taking a step in a random direction. In fact, this notion can be formalized quite nicely in the language of spectral graph theory, though this will require a short digression.

2.2 An unnatural measure of well-connectedness

Let $M \in \mathbb{R}^{V \times V}$ the adjacency matrix of G scaled by a factor of $1/D$ (we superscript by V rather than $|V|$ intentionally, to emphasize that M is to be interpreted as indexed by vertices in V). In particular, for $v, w \in V$ we can say that $M_{v,w} = E(v, w)/D$. Notice that M is symmetric because G is undirected, and that the sum of each row and column of M

is 1 (because we divided by D, the degree of each vertex). In particular, we can interpret $M_{i,j}$ as the probability of stepping from vertex i to vertex j if we choose among the D edges uniformly at random. Thus row i of M is the probability distribution over V of a length-1 random walk ('random step') from vertex i.

Suppose $\vec{e} \in \{0,1\}^V$ is a vector where entry i is 1 and the remaining entries are 0. Then $\vec{e}M$ selects row i, and is thus the probability distribution of a 'random step' from vertex i. By linearity of matrix multiplication, we can extend this to arbitrary probability distributions $\vec{\pi} \in {\mathbb{R}^+}^V$ over V, where of course $\sum_{v\in V} \vec{\pi}_v = 1$. Then, the probability distribution over V of the endpoints of a length-t random walk that begins at a vertex v chosen from distribution $\vec{\pi}$ is $\vec{\pi}M^t$.

We would like to show that for any $\vec{\pi}$, not only does $\vec{\pi}M^t$ converge to $\vec{u} = \langle 1,1,\ldots,1\rangle/|V|$ (read $\vec{u}$ as 'unit'), but does so polynomially in t. In other words, after a polynomial number of steps in a random walk, no matter where you started, your final position is uniformly random across V.

Using the Euclidean norm, let $\|\vec{\pi} - \vec{u}\|$ represent how 'close' $\vec{\pi}$ is to the uniform distribution. Similarly $\|\vec{\pi}M - \vec{u}\|$ represents how 'close' $\vec{\pi}M$ (i.e. position after a step) is to the uniform distribution. Then the ratio of those quantities $\|\vec{\pi}M - \vec{u}\|/\|\vec{\pi} - \vec{u}\|$ is a natural measure of 'how much closer' we are after a step. The lower this ratio, the faster we are converging to a uniform distribution. Define $\lambda(G) = \max_{\vec{\pi}} \|\vec{\pi}M - \vec{u}\|/\|\vec{\pi} - \vec{u}\|$, the 'worst-case' ratio. We want $\lambda(G)$ to be low for our convergence to be fast in all cases. We define $\gamma(G) = 1 - \lambda(G)$ to have a more natural quantity, the *spectral gap*, which increases as the 'connectedness' of G increases.

Definition 2.2. *A graph G has* spectral expansion γ *if* $\gamma(G) \geq \gamma$.

Though we will not prove the following facts, the motivation introduced above should render them unsurprising.

1. G is a $(|V|/2, 1+\gamma(G))$ expander.
2. A D-regular $(|V|/2, 1+\delta)$ expander has spectral expansion $\Omega((\delta/D)^2)$.

2.3 Computing $\gamma(G)$

While the expansion parameters (K, A) in the combinatorial definition are hard to compute, the expansion parameter γ in the linear-algebraic definition has a nice representation.

Recall that an *eigenvector* of a matrix M is a vector $\vec{e}$ such that $\vec{e}M = \lambda M$ for some $\lambda \in \mathbb{R}$. Here λ is called the *eigenvalue* of the eigenvector, and it is no coincidence that it shares the symbol used by the convergence ratio of the previous section...

Some elementary eigenfacts: a matrix can have multiple eigenvectors $\vec{e_1}, \ldots, \vec{e_{|V|}}$ and eigenvalues $\lambda_1, \ldots, \lambda_{|V|}$, though some eigenvectors can share an eigenvalue, leading to an eigenvalue with a *multiplicity*. Furthermore, for a symmetric matrix like M, the eigenvectors span $\mathbb{R}^V$, forming a basis.

Because the eigenvectors form a basis, we can represent $\vec{\pi}$ as a linear combination of (unit) eigenvectors, $\vec{\pi} = \sum_{1\leq i\leq |V|} \vec{e}_i c_i$. Notice that we already know that $\vec{u}M = \vec{u}$ because a random step from a uniformly random vertex leads to a uniformly random vertex (another way to see this is to note that $\vec{\pi} \cdot \vec{u} = 1$ for all probability distributions $\vec{\pi}$, since the sum of elements of $\vec{\pi}$ is 1 and there are $|V|$ elements, and the dot product with $\vec{u}$ simply adds $1/|V|$ of each element). This shows that $\vec{u}$ is an eigenvector with eigenvalue 1. It can be shown that all eigenvectors of M have magnitude between 0 and 1.

Hence, we can say $\vec{\pi} = \vec{u} + \sum_{2\leq i\leq |V|} \vec{e}_i c_i$, where we know $c_1 = 1$ because $\vec{\pi} \cdot \vec{u} = 1$ as argued above for any probability distribution vector. Suppose we multiply this equation with M. The result is $\vec{\pi}M = \vec{u}M + \sum_{2\leq i\leq |V|} \vec{e}_i c_i M = \vec{u} + \sum_{2\leq i\leq |V|} \vec{e}_i c_i \lambda_i$, by the eigenvector property. In general, then, we have $\vec{\pi}M^t = \vec{u} + \sum_{2\leq i\leq |V|} \vec{e}_i c_i \lambda_i^t$. Because all λ_i are between 0 and 1, this is ultimately dominated by the term whose coefficient is λ_2^t, the powers of the *second-largest eigenvalue* — smaller eigenvalues simply vanish much faster as a function exponential in t.

Now, recalling that $\lambda(G) = \max_{\vec{\pi}} \|\vec{\pi}M - \vec{u}\| / \|\vec{\pi} - \vec{u}\|$, we can substitute the expression for $\vec{\pi}$ above, and conclude that $\lambda(G)$ is bounded by λ_2. This gives a very natural definition of 'expansion' as $\gamma(G) = 1 - \lambda_2$, where λ_2 is the second eigenvalue of M.

3 Constructing Expander Graphs

Having defined the properties that expander graphs must satisfy, the natural follow-up question is: "How do we construct graphs with these properties?" Unfortunately, the task of explicitly constructing such a graph is pretty difficult. However, somewhat paradoxically, coming upon an expander graph accidentally is quite easy. We will explore some approaches to constructing expander graphs in this section.

3.1 Warm-up: random graphs are (probably) expander graphs

As mentioned above, while it can be very complicated to explicitly construct an expander graph, if we choose a random graph, it will be a good expander with high probability. This intuitively follows if we consider the random-walk definition of expander graphs, because it makes sense that when performing a random walk on a random graph, it is very likely that after only a small number of steps, the distribution of end-locations of all possible random walks will be near uniform. This can be proved more precisely for specific expander constants via the probabilistic method.

3.2 Expansion by squaring

So, if we pick a random graph, there is a high probability that it will have good expansion. This means that expander graphs are actually quite abundant within the space of all possible graphs. However, this still doesn't quite help us pin down an explicit construction of an expander. Instead, we turn to another question: If we have a small (constant size) graph that is a known expander, can we use it to create another, larger expander graph? The answer is yes, and this approach to expander construction was first presented by Omer Reingold, Salil Vadhan, and Avi Wigderson in 2001. Their approach focuses on two main operations: Graph Squaring and the Zig Zag Product. We will first introduce graph squaring.

Definition 3.1. *Given a regular directed graph $G = (V, E)$ of degree d, the* square *of G, denoted $G^2 = (V, E')$ is a graph with the same vertex set and additional edges such that $(u, v) \in E'$ if v is reachable from u in G with a path of length at most 2.*

Or, equivalently, we can also express graph squaring in terms of adjacency matrix multiplication as follows:

Definition 3.2. *Given a regular directed graph $G = (V, E)$ of degree d with adjacency matrix M, the* square *of G, denoted $G^2 = (V, E')$ will be equal to the d^2-regular graph with adjacency matrix $M' = M + M^2$.*

We note that by these definitions, G^2 will be a regular graph of degree d^2. By squaring a graph G, we effectively replace each random step of length 2 with a step of length 1, and thereby increase G's expansion because we can reach more vertices in a single step. Actually, we can be even more precise. Recall the eigenvalue-based definition of expansion. Clearly the eigenvalues of M^2 are λ_i^2 (each eigenvector gets scaled *twice* with two applications of the matrix), and hence $\lambda(G^2) = \lambda(G)^2$. This means $\gamma(G^2) = 1 - (1 - \gamma(G))^2 = \gamma(G)^2 - 2\gamma(G)$. It is easy to verify that this quantity is indeed greater than $\gamma(G)$, knowing that it is bounded between 0 and 1.

So, this sounds great! If we have a small expander graph G, we can just square it and increase its expansion. However, this comes at a cost. As mentioned above, the square of a regular graph of degree d will have degree d^2, and so as we increase connectivity in our graph, the degree deteriorates and we lose the sparseness of our original graph. This means that squaring alone cannot be the answer to our problem. However, we now consider a new graph operation that can help us preserve sparseness in our new graph, namely the zig-zag product.

3.3 Recovering our losses with the zig-zag product

Intuitively, what the zig zag product accomplishes is that given a larger graph and a smaller graph, it can output a graph about the size of the larger input graph but with a degree close to that of the smaller graph, all while not losing too much of the input graphs' original expansion properties. This is useful because via iterated use of graph squaring and the zig-zag product, we will be able to enlarge small constant size expander graphs to get expanders of any size.

Given a large d-regular graph $G_1 = (V_1, E_1)$ and a smaller d_2-regular graph $G_2 = (V_2, E_2)$ such that $|V_2| = d$, the zig zag product, denoted $G_1 \textcircled{z} G_2$, is a graph on the set of vertices $V_1 \times V_2$. This can be thought of as replacing each $v \in G_1$ with a copy of the vertices G_2. We will call each of these copies a "cloud." The degree of G_1 is equal to the size of the cloud, so we now have one vertex in the cloud for every edge leaving the original vertex v in G_1.

Very generally, starting from any vertex in this new graph, we will be adding edges to $G_1 ⓩ G_2$ if we can make one step within the starting vertex's cloud (along one of the original edges in the smaller graph), then a step between clouds (along one of the edges in G_1), and then make a third step within the new cloud (again along the edges of the smaller graph G_2) to reach the end vertex of the new edge. This is not the exact zig-zag product, because there is a more explicit definition of exactly which edges we can traverse, but this is the basic intuition which we will elaborate on and formalize very soon.

This way of obtaining edges is where the zig-zag product gets its name, because we are alternating small steps within clouds and a larger step between clouds, resulting in a somewhat zig-zag-like movement. Now that we have an intuition for the basic structure of edges in the zigzag product of two graphs, we can define it formally. Suppose that we have the graphs as described above, but we give some sort of fixed labeling to the neighbors of each vertex, so that for some $v \in G_1$, $v[i]$ refers to the neighbor of v labeled i, with the same being true for G_2. Using this labeling, we have the following definition:

Definition 3.3. *Given a large regular graph $G_1 = (V_1, E_1)$ with degree d_1, and a small regular graph $G_2 = (V_2, E_2)$ with degree d_2 and $|V_2| = d_1$, we define the zig zag product, denoted $G_1 ⓩ G_2$ to be a graph with the vertex set $V_1 \times V_2$ and edge set E such that for vertices $v_1, v_2 \in G_1$ and $v_3, v_4 \in G_2$, we have $((v_1, v_3), (v_2, v_4)) \in E$ if and only if there exists some $(i, j) \in [d_2] \times [d_2]$ such that $(v_1[v_3[i]], v_3[i][j]) = (v_2, v_4)$.*

Plainly, a formal zig-zag step consists of starting from some vertex (i, j) in $V_1 \times V_2$, choosing some j'th neighbor of j in G_2 which can be thought of as having a label between 1 and d_1 because $|V_2| = d_1$, we can call it c, so we are now at the vertex (i, c). We now move to the cth neighbor of i in G_1, called i' so we are now at (i', c). This is the long step of the zig-zag (between clouds). We then finish with a short step from c to any of its neighbors in G_2, c', ending at some (i', c'), and completing a zig zag step. So, for every (i', c') that can be reached from (i, j) in this way, the zig zag product will have an edge between (i, j) and (i', c').

So, we have our zig-zag product, but why does this give us the properties that we want, i.e. a larger graph with reduced degree but still relatively good expansion? We consider what it means to take a random walk on $G_1 ⓩ G_2$. By the definition of the zig zag product, taking one random step on $G_1 ⓩ G_2$ corresponds to taking two independently random steps on G_2 (i.e. the steps within the start and end clouds). So, we are able to move toward a uniform distribution on our graph about as quickly as we move toward a uniform distribution on G_1 and G_2, thereby keeping expansion properties about the same, while enlarging the graph at the same time.

3.4 Putting all the pieces together for a (roughly) explicit construction

So, we now have an operation that can improve the expansion of a graph while preserving the side and increasing the degree (squaring), and another operation that can increase the size of a graph keeping the expansion and degree relatively constant (the zig zag product). So, we can put them together to define an operation that will enlarge a graph while preserving its expansion properties. When we perform this operation multiple times, we can grow a small expander graph into a larger expander graph, giving us an explicit construction for expander graphs. We present the following iterative definition for this process:

Definition 3.4. *Given a d-regular graph H with d^4 vertices and very good expansion ($\geq 7/8$), we define the sequence of graphs $G_1, G_2, G_3, ...$ as follows:*

$$G_1 = H^2$$

$$G_t = G_{t-1}^2 \, ⓩ \, H$$

From the properties of squaring and the zig-zag product, it is intuitive that this would give us a sequence of graphs increasing in size, the same degree as H^2, and relatively good expansion. Specifically, with some exact analysis of how expansion is affected by the zig-zag product, which we have omitted here, we have

Lemma 3.5. *Given the sequence of graphs as defined in Definition 3.4, for every $t \geq 1$, G_t will be a d^2-regular graph on d^{4t} vertices with good expansion ($\geq 1/2$).*

Now that we have a way to construct expander graphs, we will investigate how they are used in the proof that undirected connectivity is in **L**, as presented by Omer Reingold in his 2004 paper.

4 Using Expander Graphs to Derandomize USTCON

One of the most powerful applications of expander graphs has been their use in the proof that undirected connectivity can be computed in logspace. We begin our overview of the proof with a reminder of the problem statement for undirected connectivity, usually denoted as USTCON.

Definition 4.1. USTCON $= \{\langle G, s, t \rangle | G$ *is an undirected graph that contains vertices s and t such that G contains a path from s to t*$\}$

A few well-known facts about USTCON readily follow from its definition.

4.1 A Few Immediate Facts about USTCON

Theorem 4.2. USTCON $\in$ ***P***.

Proof. This quickly follows from applying any of the well-known polynomial-time path-finding algorithms such as breadth-first-search. $\square$

Slightly less obviously, we also have

Theorem 4.3. USTCON $\in$ ***RL***.

To prove this, we need the following lemma:

Lemma 4.4. *Given a graph $H = (V, E)$ and two vertices $s, t \in V$ that are connected in H, the expected time for a random walk from s to get to t is at most $2|E|$.*

Using this lemma without proof, we now have

Lemma 4.5. *Given a graph $H = (V, E)$ and two vertices $s, t \in V$, the probability that we reach t at some point in a random walk from s of length $2|E|$ is*

$$Pr[\textit{we reach t from s}] \begin{cases} > \frac{1}{2} & \textit{if s and t are connected.} \\ = 0 & \textit{otherwise} \end{cases}$$

Proof. Clearly, if s and t are not connected, we will never reach t in a random walk from s, and so the probability is 0 as stated.

Otherwise, by Lemma 4.4, we know that the expected time to get from s to t is at most $2|E|$, and so if we perform a random walk for $2|E|$ steps, the probability that we reach t at some point during that walk must be more than $1/2$. $\square$

Using this, we can complete our proof that USTCON $\in$ **RL**:

Proof. it is clear that USTCON $\in$ **RL** because given any graph H, we know that the number of edges is at most n^2, and so if we perform a random walk from s for $2n^2$ steps and output *yes* if at any point we reach t, and *no* if we never encounter t, by the above lemma (4.5) we know that

$$Pr[\text{output yes}|s \text{ and } t \text{ not connected}] = 0$$

$$Pr[\text{output yes}|s \text{ and } t \text{ connected}] > \frac{1}{2}$$

and to run the algorithm, we need only keep track of the current vertex we are at, and the number of steps we have taken, which will require

$$O(\log(2n^2)) = O(\log(n))$$

space, and so this algorithm places USTCON in **RL**. $\square$

This notion that we can place USTCON in **RL** through the use of random walks plays a large part in the proof that USTCON can in fact be de-randomized and placed in **L**.

4.2 Intuition for De-randomization

After seeing the simple randomized algorithm above that can compute USTCON in log-space via the use of a random walk, the natural follow up question is whether we could somehow apply similar techniques to place USTCON in deterministic log-space.

We note that in any graph on n vertices, there exists a path between any pair of connected vertices with length at most n. This is because any path longer than n would have to visit at least one of the vertices in the graph multiple times, which would imply that the path had a loop, and we could just remove the loop to reduce the length of the path.

So, instead of randomly trying a path and hoping that it works, we could de-randomize our algorithm by trying *all* paths of length n from our starting vertex, s, and accepting if we encounter the end vertex, t, at some point. By the above, we know that if s and t are connected, one of the paths must contain t and so we will always accept, and if s and t are not connected, there is no path from s to t and so we will always reject. Therefore, this algorithm will always be correct, and therefore solves USTCON deterministically.

If we assume our graph has degree d, where d is some constant, then the total number of paths we have to search would be d^n, because at each step of the path we have d neighbors to choose from. In order for our algorithm to systematically search every possible path, we would need to enumerate these paths, which will take $O(\log d^n) = O(n \log d) = O(n)$ space. However, that means that this algorithm uses a bit too much space compared to what we want, because we are looking for a way to place USTCON in logspace. To do this, we will have to modify our approach a bit.

4.3 Adjusting our Input Graph to Make it More Accomodating

So, our de-randomized approach to USTCON seems promising, but as we saw in the previous section, if we just apply it to our input graph as-is, we aren't able to place it in log-space. However, what if our graph was *more* connected? What if instead of requiring $O(n)$ steps to reach t from s in our d-regular graph with high probability, we only required $O(\log(n))$? It seems like expanders could be useful here, and in fact from their definition, we know that

Lemma 4.6. *Given any d-regular $(\frac{n}{2}, \epsilon)$ expander graph G over n vertices where ϵ is some constant greater than 1, for any two connected vertices s, t of G, there exists a path of length $O(\log(n))$ from s to t.*

Proof. This follows directly from our definition of expansion, because for every graph with $(\frac{n}{2}, \epsilon)$, expansion, then for any subset of vertices S of size at most $n/2$, the number of edges between that subset and the rest of the graph is at least $\epsilon \cdot |S|$. So, by iteratively applying this definition, we see that there exists some $k = O(\log(n))$ such that there are at least $n/2$ vertices within k steps of s, and $n/2$ vertices within k steps of t, and so these two sets must overlap, giving us some vertex v that is at most k steps from both s and t, and so we have a path from s to t through v that takes at most $2k = O(\log(n))$ steps. □

So, if our graph was an expander, we would only have to enumerate over all paths of length $O(\log(n))$, instead of all paths of length $O(n)$, which would allow us to compute USTCON for such graphs in logspace, because the number of such paths would be $d^{O(\log(n)}$, which could be enumerated using $O(\log(d^{\log(n)}) = O(\log(n)\log(d))$, which is equal to $O(\log(n))$ if d is constant.

So, how might we transform our graph into a better expander graph? Well, we have already explained earlier how the zig-zag product and graph squaring can be used in tandem to improve the expansion of a graph while keeping the degree constant. We also notice that both of these operations preserve the connected components of the input graphs in their output. So, while this must be stated more explicitly for a rigorous proof, here we will claim that using iterated applications of graph squaring and the zig-zag product (where the second input to the product is an expander of constant size), we can transform our input graph H into a graph of constant degree, where every connected component is an expander graph. Furthermore, we claim this can be done in log-space. Using this fact, we now have all of the results required to prove our goal.

4.4 Putting Together all of the Pieces

Now that we have considered all parts of the complete proof that USTCON $\in$ **L**, the full proof quite elegant, and consists of an algorithm that depends on the de-randomization properties of expander graphs and graph transformation via squaring and the zig-zag product. We present an overview of the proof below:

Theorem 4.7. *USTCON $\in$ **L**.*

Proof. This can be proved by defining an algorithm that brings together the various results we have looked at in this section, and then showing that at each step the algorithm uses at most logspace. We define the algorithm as follows:

```
On input $\langle G, s, t \rangle$ :
1) Using a combination of graph squaring and the zig zag product,
   transform $G$ into a graph $G'$ that has constant degree $d$, and
   each component is an expander.
2) Enumerate all paths of length $O(\log(n))$ from $s$ in $G'$.
3) Check each path. If any path contains $t$, output *yes*.
4) Otherwise, output *no*.
```

We claim that step 1) can be done in log-space, and then there will be $d^{O(\log(n))}$ paths of length $O(\log(n))$, and so all such paths can be enumerated in space $O(\log(d^{\log(n)})) = O(\log(n) \cdot \log(d))$, which is equal to $O(\log(n))$ because d is a constant. So, step 2) can also be accomplished in log space. In order to check a specific path, we need only remember along which edges we've traveled so far in the path. Because each path has length

$O(\log(n))$, this means that the information required for steps 3) and 4) can also be stored in log space.

So, overall, this is an algorithm where each step requires at most log-space to compute, and so the entire algorithm runs in log-space. If we assume that this algorithm is correct due to what we have studied in previous sections, we can conclude that the algorithm can compute USTCON in logspace. $\square$

5 Conclusion

The past few pages have been a bird's-eye view of Reingold's proof that USTCON $\in$ **L** (we refer readers interested in descending further into details to Salil Vadhan's notes and Reingold's original paper, both of which are listed in the references below).

Why is this result so celebrated? Why should we care that USTCON $\in$ **L**? First, as with many problems in complexity theory, showing that USTCON can be solved in logarithmic space implies that many other problems that *logspace-reduce* to USTCON can also be solved in logarithmic space. Examples of such problems are counting connected components and testing for bipartite-ness. Reingold's result places them in **L** in one fell swoop.

The big picture is even more exciting: Reingold's theorem is a natural step on the way to proving the conjecture that **RL** = **L**, which would imply that *all* randomized algorithms that can be computed in logarithmic space can be de-randomized in logarithmic space. In other words, this would imply that the power of randomness does not help you save on memory usage!

The crucial point here is that to make claims about **RL**, we must reason about *directed* rather than *undirected* graphs as Reingold's theorem does; this is because not all computations are reversible. In his paper, Reingold notes that "[w]hile progress in the context of RL does not seem immediate ... we feel that it is still quite plausible." So there is hope — and studying expander graph-related techniques further may well be the key to showing **RL** = **L**.

Finally, on a personal note, we [the authors; two second-year undergrads] found this proof exceptionally elegant and satisfying to explore. In many ways it is a true gem of complexity theory, combining a wide variety of fascinating tricks, and for that reason alone we think it is worth studying and celebrating.

References

[1] Shuchi Chawla, "Lecture 15 Notes: Random Walks and Markov Chains", *CS787: Advanced Algorithms*, University of Wisconsin, Madison, Fall 2009. http://pages.cs.wisc.edu/ shuchi/courses/787-F09/scribe-notes/lec15.pdf

[2] Emmanuel Kowalski, "An introduction to expander graphs", *ETH Zurich*, August 2018. https://people.math.ethz.ch/ kowalski/expander-graphs.pdf.

[3] Omer Reingold, Salil Vadhan, Avi Wigderson, "Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors". *Ann. of Math*, 155(1), 157-187, 2002. arXiv:math/0406038.

[4] Omer Reingold. 2008. "Undirected connectivity in log-space". *J. ACM* 55, 4, Article 17, September 2008. http://dx.doi.org/10.1145/1391289.1391291.

[5] Salil Vadhan, "Chapter 4: Expander Graphs", *Foundations and Trends in Theoretical Computer Science* : Vol. 7: No. 13, pp 80-127, now publishers, December 2012. https://people.seas.harvard.edu/ salil/pseudorandomness/.